IB 03/2720

Rec'd PCT/PTO 27 DEC 2004

IB 03/2720

PRV

PATENT- OCH REGISTRERINGSVERKET

Patentavdelningen

**Intyg**
**Certificate**

*Härmed intygas att bifogade kopior överensstämmer med de handlingar som ursprungligen ingivits till Patent- och registreringsverket i nedannämnda ansökan.*

*This is to certify that the annexed is a true copy of the documents as originally filed with the Patent- and Registration Office in connection with the following patent application.*

PATENT- OCH REGISTRERINGSVERKET

| | | |
|---|---|---|
| *(71)* | *Sökande* / *Applicant (s)* | *ABB AS, Billingstad NO* |
| *(21)* | *Patentansökningsnummer* / *Patent application number* | *0202019-6* |
| *(86)* | *Ingivningsdatum* / *Date of filing* | *2002-06-28* |

REC'D 08 OCT 2003
WIPO PCT

*Stockholm, 2003-09-30*

*För Patent- och registreringsverket*
*For the Patent- and Registration Office*

Kerstin Gerdén

*Kerstin Gerdén*

*Avgift*
*Fee 170:-*

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

2002-05-14
4-9432SE/LG

# Revalidation of a compiler for safety control

## TECHNICAL FIELD

The present invention concerns revalidation of a compiler of control language for use in an industrial control system. In particular the invention reveals a method to revalidate a compiler for compilation of a user written program, which is intended for safety control of real world entities. The user written program subject to compilation by the compiler is intended for execution in a device, which comprises functionality that adds safety features to an industrial control system. The invention insures that no fault is introduced into the device due to error in the compiler code. Such an error can for instance occur during distribution of the compiler code. An error can also occur due to failure in a computer's memory or on a disk where the compiler code is stored. The invention insures that no such fault is introduced into the control of real world entities which otherwise could lead to accidents that harm people or cause damage to the environment.

## BACKGROUND ART

Industrial control systems are for instance applied in manufacturing and process industries, such as chemical plants, oil production plants, refineries, pulp and paper mills, steel mills and automated factories. Industrial control systems are also widely used within the power industry. Such industrial control systems may need to comprise or be combined with devices, which adds safety features. Example of processes which requires additional safety features than what a standard industrial control

PRV 02-06-28

system provides are processes at off-shore production platforms, certain process sections at nuclear power plants and hazardous areas at chemical plants. Safety features may be used in conjunction with safety shutdown, fire and/or alarm systems as well as for fire-and-gas detection.

An example of an industrial control system, which includes a safety critical function, is described in DE19857683 "Safety critical function monitoring of control systems for process control applications has separate unit". The system has a main controller bus coupled to different processors via a number of decentralized data receivers.

The use of general-purpose computer systems raises issues in that a user written program do not get affected by a fault in the compiler code.

"Compilers: Principles, techniques and tools" by Alfred V. Aho, Ravi Sethi and Jeffrey D. Ullman published 1988 by Addison-Wesley publishing company, includes a discussion on verification of general-purpose compilers. Page 731 paragraph 11.4 "Testing and maintenance" handles the verification of compilers according to standard software testing. One approach, suggested in the book, is the "regression" test. A suite of test programs is maintained, and whenever a compiler is modified, the test programs are compiled using both the new and old version of the compiler. Any difference in the target programs produced by the two compilers is reported to the compiler writer. Further the book points out that choosing the programs to include in a test suite is a difficult problem.

Prior art in the area of compilation technology include methods and systems for compiler optimization. US5577253 "Analyzing inductive expressions in a multilanguage optimizing compiler" describes a method executed in a computer system where a plurality of optimizations is performed by a generic compiler back-end using induction variables. This patenting optimization technique does not address the correctness of a compiler at a later time.

US6071316 "Automated validation and verification of computer software" shows a method for verifying that a source code, which has been compiled, executes all different paths in the code. This is not concerned with the compiler correctness.

A remaining problem in the area of safety control of real world entities is to insure the highest possible reliability of a user written program.

Another problem relating to industrial control systems is that the complexity of system software distribution (such as via Internet) has led to an increased risk of that errors occurs in the compiler software.

The inventors have found that there is a need to ensure that a compiler for software with the purpose of safety control of real world entities do not change it's way to produce code during that it is distributed, stored as binary code or is recompiled from the associated high level code.

## SUMMARY OF THE INVENTION

An object of the present invention is to provide a method to revalidate a compiler intended for compilation of a user written program for execution of safety control in an industrial control system.

This and other objects are fulfilled by the present invention according to a method described in a claim 1. Advantageous embodiments are described in sub-claims.

With the present invention a test program, defined in a control language, is compiled. By verifying that the test program executes correctly, the compiler is validated. A first software means for later comparison purposes is generated. In conjunction with the compilation of a user written program the test program is compiled. Based on the second compilation of the test program a second software means is generated. The compiler is revalidated for errors introduced between the first and second compilation by comparing the first and second software means. Provided that the revalidation indicates no errors in the compiler, the user written program is enabled to execute in a device with safety features for control of real world entities.

The user written program subject to compilation by the compiler is intended for execution in a device, which comprises functionality that adds safety features to an industrial control system. As mentioned above a method according to the invention include steps which shows how to generate a first and second software means based on a compiled test program. Typically the first software means is generated at the time of establishing of new version or revision of the compiler of a control language. The first software means is typically associated with the revision or version of the compiler code at hand. The method comprises steps where a second software means is generated at the time of compilation of a user written program. The method comprise steps in which the first and second software means are used to revalidate the compiler by comparing the first software means with the second software means. The first software means and the second software means are derived from the compiled test program by use of the same principals.

The invention facilitates that no fault is introduced into the industrial control system due to error in the compiler code. Such an error can for instance occur during distribution of the compiler code or an error can be due to failure in a computer's memory or failure in a disk where the compiler code is stored. An error in the compiler code can also occur due to faults in a computer register, a stack memory or in a CPU.

A particular useful feature of the invention is that it facilitates that no such fault is introduced into the device for safety control of real world entities which otherwise could lead to accidents that harm people or cause damage to the environment.

The user written program is typically written in control language, for instance based on IEC 61131-3.

An aim of the invention is to detect a fault in the compiler code. The invention detects errors in the

PRV 02-05-23

compiler code at any time of compilation, which insures a high reliability of safety critical user written program compiled by the said compiler.

A further object of the invention is to provide a computer program product containing software code means loadable into the internal memory of a general-purpose computer or workstation and/or a device, which computer program products has software means to execute at least one step of the above described method.

Yet a further object of the invention is to provide a computer program comprising computer code means and/or software code portions for making a computer or processor perform any of the steps of the above described method.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be described in more detail in connection with the enclosed schematic drawings.

Figure 1 shows a schematic overview of an industrial control system comprising a computer loaded with compiler code and a device with safety control features.
Figure 2 shows a schematic flowchart of a method based on the invention.
Figure 3 shows a simplified diagram of an embodiment of the invention in where the compilation of the test program is performed at the same time as the compilation of a user written program. The compiled test program is compared with a previous compilation of the test program.
Figure 4 shows a simplified diagram of another embodiment of the invention where the compilation of the test program is performed at the same time as the compilation

of a user written program. The figure shows that the second software means is downloaded to a device executing safety control where it is compared with the first software means.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 shows a schematic diagram of an industrial control system 2 with a device 6a comprising safety features 6b. A user written program intended for safety control of real world entities 10 is typically compiled in a workstation 5a or in a general purpose computer. Such a workstation 5a or general purpose computer is connected by communication means 3 to the device 6a. The communication means 3 is based on communication standards such as fieldbus technology or such as TCP/IP. The industrial control system 2 comprise a multitude of different devices such as controllers 6c, PLCs 7, operator stations or process portals 4 and process I/O 8. The above mentioned devices may exist in any number and in combination with other devices common in an industrial control system. The device 6a comprising safety features 6b may be an individual device such as a PLC or controller. The safety features are such that the device and/or the industrial control system comply with safety standards such as Safety Integrity Levels (SIL) as defined in the standard IEC 61508. The device may also comprise one or several software modules with safety features added to the device. The device 6a is connected to real world entities 10 subject to safety control via communication means such as a fieldbus or process I/O. Examples of real world entities are actuators, instruments, motors, valves, pumps, fans etc. A real world entity may also be a group of entities or a system of entities.

A device 6a for safety applications in a process control system 2 typically execute user written applications described in a high level language derived from the standard IEC 61131-3, which is well known to a person skilled in the art. Hence, the compiler 22 is typically a compiler for a high level language derived from the standard IEC 61131-3.

Hereafter a release, a version or a revision of the compiler is called the compiler.

Validating a compiler for safety control is typically made at a software factory. A software factory is in this context a location where sufficient and certified test equipment as well as qualified personal is available to perform tests and validation of the compiler. Validation of the compiler and the associated tests should be substantial. The tests should for instance insure that the compiler 22 and the safety features meet requirements of safety certification. Also other requirements need to be met such as sufficient performance in order for other applications or programs to execute in the industrial control system 2. The validation of the compiler comprise verification of that applications executes correctly in the device for safety control of real world entities.

The invention disclose that, in addition to the above described validation of a compiler, a test program 20 is established where the purpose of the test program 20 is to use it as input for revalidation of the compiler 22 outside the software factory. A test program 20 should include all logic of the control language, which is used for safety control applications. The definitions used in a typical test program are typically derived from the IEC 61131-3 standard. A preferred test program is built by using all languages, all functions and all language constructs. This in order to insure that the compiler 22 parses and checks all logic expressions during compilation of the test program which later are to be used in a user written program 21.

In an embodiment of the invention a version or revision of the compiler from the software factory is associated with the test program. The test program is at least partly used in the validation of the compiler at the software factory. A first software means intended for later comparison purposes is also associated with the version or revision of the compiler. It is further advantageous to distribute the test program together with the release, version or revision of the compiler.

Figure 2 shows a schematic flow chart of a method based on the invention. The test program 20 is defined in a control language. The method comprises the step of compiling 11a the test program by means of the compiler. Further the method comprises the step of validating 11b the compiler by verification of that the test program executes correctly.

Figure 2 also shows that the method comprises the step of generating a first software means 12. The first software means is dependant of the executable code of the test program 20. The first software means may have many embodiments. In one embodiment the first software means 23 comprise the original executable code of the compiled test program. In another embodiment generating 12 a first software means comprise the calculation of a check-sum and/or a code for cyclic redundancy check. In such an embodiment the check-sum and/or the code for cyclic redundancy check is calculated with the compiled test program as one input. Hereafter a code for cyclic redundancy check is called CRC. A CRC can be calculated or derived in several ways. For instance, a CRC can be of length 16 bit or 32 bit. The 16 bit polynomial CRC-CCITT $(X^{16}+X^{12}+X^{5}+1)$ or the 16 bit polynomial $(X^{16}+X^{15}+X^{2}+1)$ are example of polynomials suitable to be used in embodiments of the invention. An example of a 32 bit polynomial, which can be used to calculate a CRC, is $(X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1)$. The mentioned 32 bit polynomial is defined in the Ethernet standard IEEE 802.3 and is a preferred polynomial to use in embodiments of the invention. In an alternative embodiment a checksum, such as a parity check, could be used.

Figure 2 shows that a method based on the invention comprises the step of compiling 13 the test program 20 a second time. Compiling 13 the test program 20 is made in conjunction with the compilation of a user written program 21. The time lag between generating 12 the first software means 23, 35 and compiling 13 the test program 20 a second time, is typically, several days or weeks. The time lag may be up to several years. During the time between generating 12 the first software means and compiling 13 the test program a second time, an error in the compiler code might have occurred. Such an error can for instance occur during distribution of the compiler code or an error can be due to failure in a computer's memory or failure in a disk where the compiler code is stored. An error in the compiler code can also occur due to faults in a computer register, a stack memory or in a CPU.

Further, figure 2 shows that the method comprises the step of generating 14 a second software means 24a, 31 based on the second compilation of the test program 20. The step of generating the second software means 24a, 31 is based on the same principals as the previous step of generating 12 the first software means. As with the first software means the second software means may have many embodiments. In one embodiment the second software means 24a comprise the executable code of the second compilation of the compiled test program 20. In another embodiment generating 14 the second software means comprise a calculation of a check-sum and/or a code for cyclic redundancy check. Alternative ways of calculation a check-sum and/or a code for cyclic redundancy check are described in more detail in the above description of generating the first software means. Figure 3 shows a more detailed overview of generating 14 a second software means 24a and the following steps of comparing 15 software means and enabling 16 the user written program 26. Figure 4 indicates that in an alternative embodiment of the invention the second software means 31 is downloaded to the device 6a.

Figure 2 also shows that the method comprises the step of comparing 15 the first software means with the second software means. Figure 3 shows that in one embodiment of the method the comparing step is performed by means of the same workstation 5a or general purpose computer as in which the compiler 22 is installed. In such embodiment the comparing step 15 of the first software means 23 and the second software means 24a may be implemented by use of standard features provided by an operating system.

In another embodiment the comparing step 15 is performed by means of the device 6a. Figure 4 shows an overview of such an embodiment. In such an embodiment it is preferable that the first software means 35 is down-loaded to the device 6a together with the system software. Figure 4 indicates that the first software means 35 typically has been down-loaded to the device 6a before the second compilation of the test program 20. The second software means 31 is down-loaded 34 in conjunction with a successful compilation of the user written program 26.

Comparing 15 the software means does, in one embodiment of the invention, involve a compare of the reminder values - and not between the values where the reminder is included in the calculation. In the latter case the value will be 0 and a comparison between 0 and 0 may result in that the stored calculation is placed in a memory where some or all bits are stuck at 0 and the comparison may give an invalid result. That is why a comparison between nonzero values (such as reminder values) yields a higher probability to discover faults.

In an alternative embodiment of the invention the steps of compiling the test program 13, generating 14 a second software means and comparing 15 the first and second software means is repeated any number of times. In such an alternative embodiment an additional source of data may be used with the purpose of generating a change in both the first and second software means. An example of such an embodiment is that the generating step of the second software means comprise an additional step of combining a variable that change over time with the second software means. The variable that change over time typically relates to the second compilation of the test program. In the same alternative embodiment the comparing step may comprise an additional step of down-loading the variable that change over time. It is advantageous to use a date&time stamp. In one embodiment according to figure 4 the date&time stamp is down-loaded to the device 6a and the date&time stamp is combined with the first software means 35. The advantage of using a variable that change over time, such as a date&time stamp, is to generate a change in the first and second software means over time. Such a change eliminates the possibility that one unit in the down-load chain, stores the second software means during a down-load and at a later down-load sends that second software means instead of the new one it receives.

Figure 2 also shows that the method comprise the step of enabling 16 the compiled user written program 26 to execute in the device 6a with safety features for control of real world entities 10. The enabling step of the method is performed provided that no errors were detected in the compiler in the previous steps.

A method according to the invention is at least partly performed under control of a set of computer readable instructions contained in a computer program storage device.

The invention also discloses a computer program product 5b intended for safety control in an industrial control system 2. The computer program product 5b comprise functionality of enabling a user written program to execute after revalidating the compiler according to the above described methods. Further the computer product comprises software means for carrying out a further action to receive a signal sent across internet 1 comprising the first software means 35.

The invention also discloses a computer program comprising computer code means for making a computer or processor perform any of the steps of the above described method.

The foregoing disclosure and description of the invention are illustrative and explanatory thereof, and various changes in the components, processing and computational steps and procedures, as well as in the details of the illustrated circuitry and method of operation may be made without departing from the spirit of the invention.

PRV 02-06-28

CLAIMS

1. A method to revalidate a compiler (22) intended for compilation of a user written program for safety control in an industrial control system (2), comprising the steps of
- compiling (11a) a test program (20) a first time which test program is defined in a control language;
- validating (11b) the compiler by verification of that the test program executes correctly;
**characterized** by the further steps of
-generating (12) a first software means derived from the compiled test program intended for later comparison purposes;
-compiling (13) the test program a second time in conjunction with the compilation of a user written program;
-generating (14) a second software means intended for a comparison based on the second compilation of the test program;
-comparing (15) the first software means with the second software means; wherein the compiler (22) is revalidated for errors introduced between the first and the second compilation;
- enabling (16), provided that the revalidation indicates no errors in the compiler (22), the user written program to execute in a device (6a) with safety features for control of real world entities (10).

2. A method according to claim 1, **characterized** in that the comparing step (15) is performed in the same workstation (5a) or general-purpose computer as the compiler (22) is executing.

3. A method according to claim 1, **characterized** in that the software means is a check-sum or a code for cyclic redundancy check.

4. A method according to claim 3, **characterized** in that the comparing step (15) is performed in the device (6a) with safety features.

5. A method according to claim 4, **characterized** in that the comparing step (15) comprise an additional step of down-loading a variable that change over time, which is down-loaded in the same message as the check-sum or code to the device (6a), where the variable that change over time is used to achieve a change in the message.

6. A method according to claim 1, **characterized** in that the test program (20) is defined in a control language derived from the standard IEC 6-1131.

7. A computer program product (5b) containing software code means loadable into the internal memory of a general-purpose computer or workstation (5a) and/or a device (6a), **characterized** in that said computer program product has means to execute a computer implemented step of compiling (13) the test program a second time, a computer implemented step of generating a second software means (14), a computer implemented step of comparing (15) the first software means with the second software means and a computer implemented step of enabling (16) the user written program to execute in the device (6a), all steps according to claim 1.

8. A computer program product (5b) according to claim 7 which comprise software means for carrying out a further action to:
-receive a signal sent across internet (1) comprising the first software means (35).

9. A computer program comprising computer code means and/or software code portions for making a computer or processor perform any of the steps of claims 1-6.

## ABSTRACT

The present invention concerns revalidation of a compiler of control language for use in an industrial control system. In particular the invention reveals a method to revalidate a compiler for compilation of a user written program, which is intended for safety control of real world entities. The user written program subject to compilation by the compiler is intended for execution in a device, which comprises functionality that adds safety features to an industrial control system. The invention insures that no fault is introduced into the device due to error in the compiler code. Such an error can for instance occur during distribution of the compiler code or due to failure in a computer's memory or failure on a disk where the compiler code is stored. Hence the invention insures that no such fault is introduced into the control of real world entities which otherwise could lead to accidents that harm people or cause damage to the environment.

Fig. 2

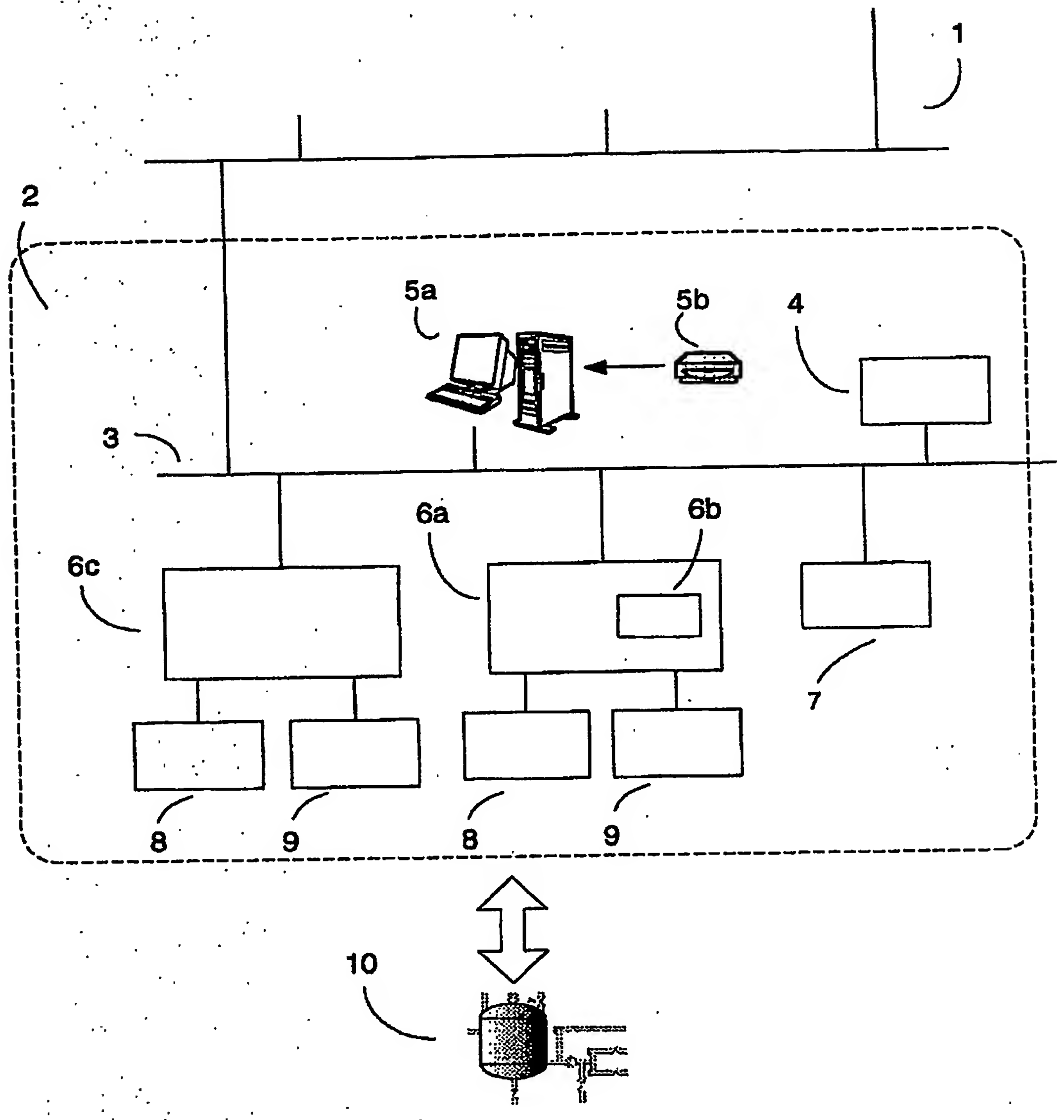

Fig. 1

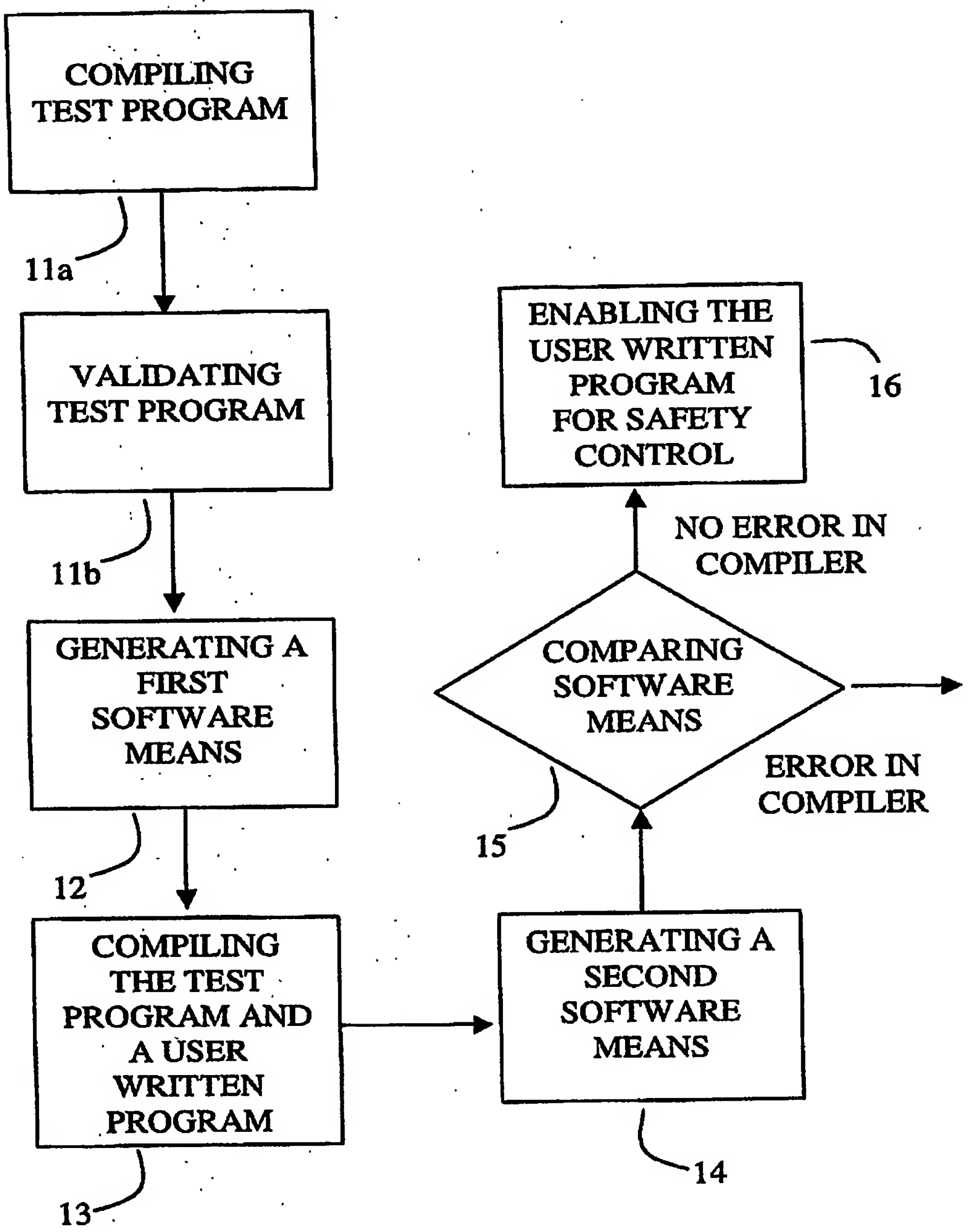
COMPILING TEST PROGRAM
11a
VALIDATING TEST PROGRAM
11b
GENERATING A FIRST SOFTWARE MEANS
12
COMPILING THE TEST PROGRAM AND A USER WRITTEN PROGRAM
13
GENERATING A SECOND SOFTWARE MEANS
14
COMPARING SOFTWARE MEANS
15
ERROR IN COMPILER
NO ERROR IN COMPILER
ENABLING THE USER WRITTEN PROGRAM FOR SAFETY CONTROL
16

Fig. 2

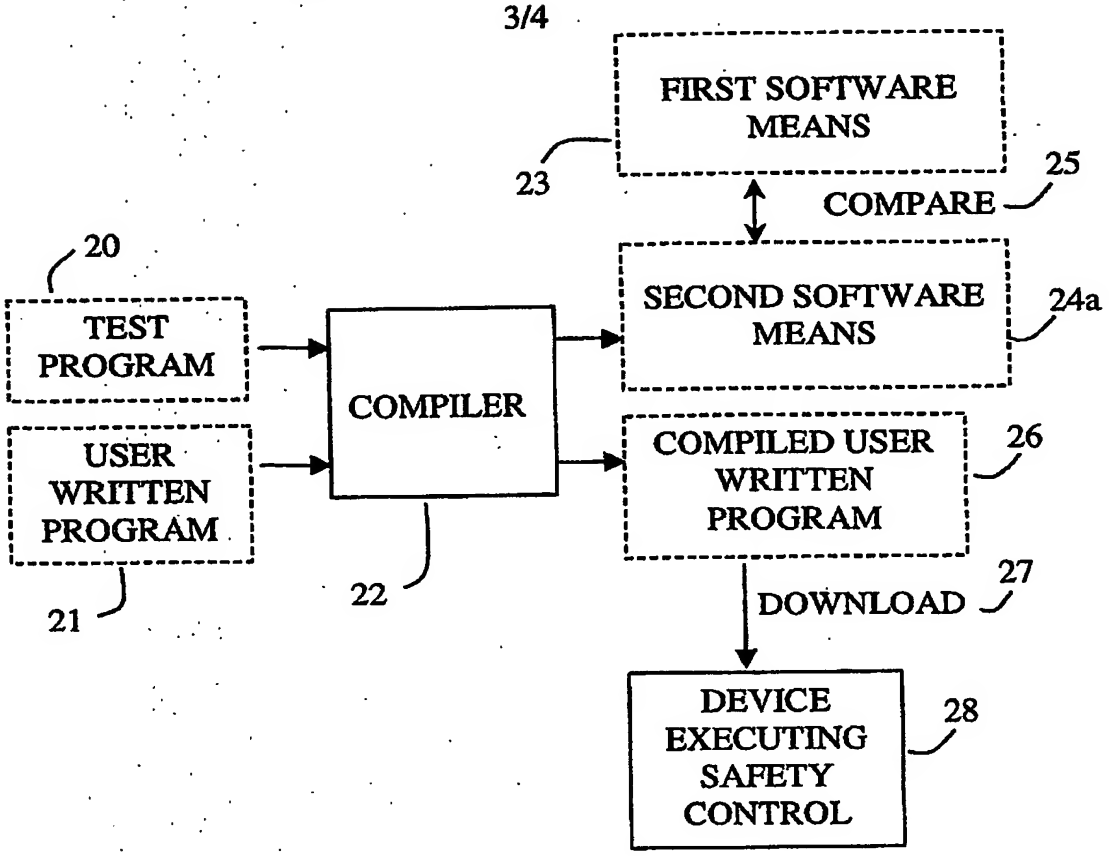
FIRST SOFTWARE MEANS
23
COMPARE
25
SECOND SOFTWARE MEANS
24a
20
TEST PROGRAM
USER WRITTEN PROGRAM
21
COMPILER
22
COMPILED USER WRITTEN PROGRAM
26
DOWNLOAD
27
DEVICE EXECUTING SAFETY CONTROL
28

Fig. 3

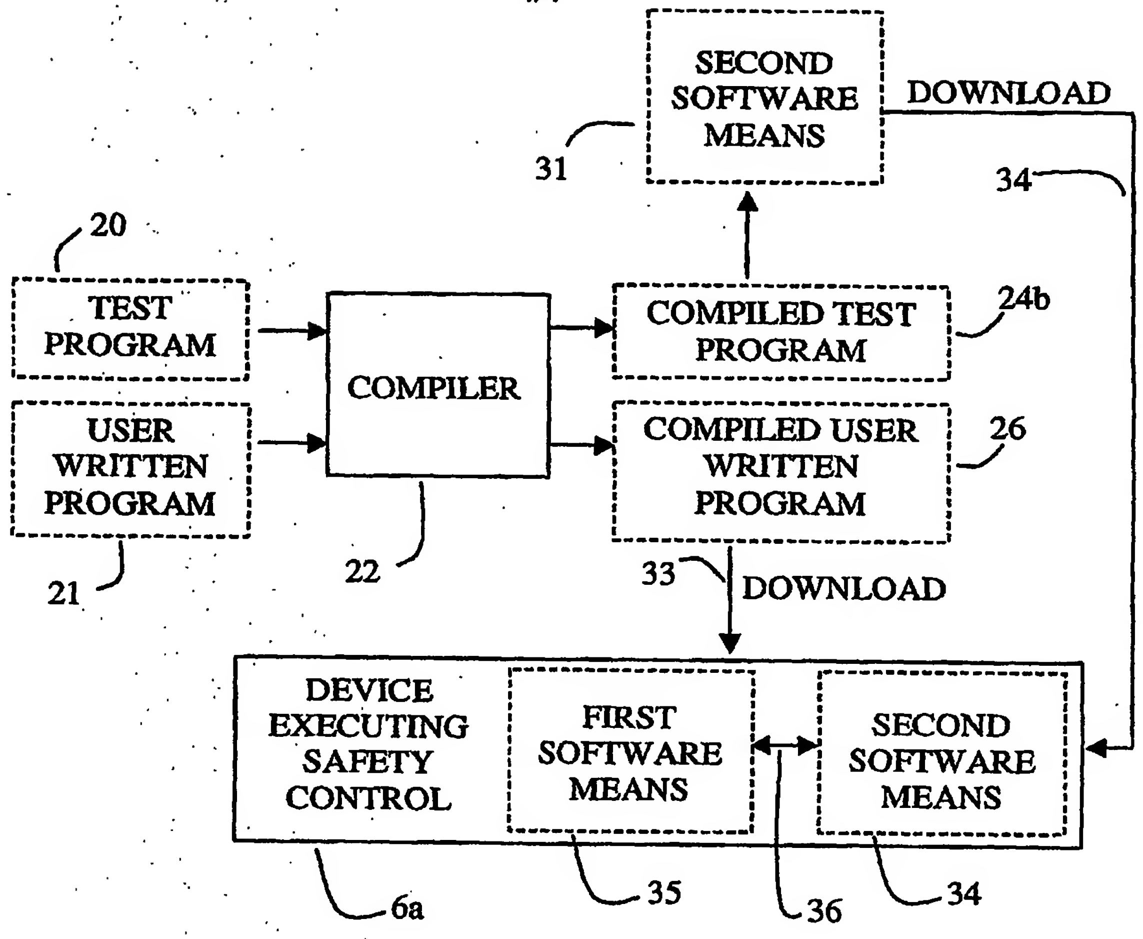
SECOND SOFTWARE MEANS
31
DOWNLOAD
34
20
TEST PROGRAM
USER WRITTEN PROGRAM
21
COMPILER
22
COMPILED TEST PROGRAM
24b
COMPILED USER WRITTEN PROGRAM
26
33
DOWNLOAD
DEVICE EXECUTING SAFETY CONTROL
FIRST SOFTWARE MEANS
SECOND SOFTWARE MEANS
6a
35
36
34

Fig. 4